

Parameter Estimation in High Dimensional Gaussian Distributions

Erlend Aune (NTNU, Norway) and Daniel P. Simpson (NTNU, Norway)

January 25, 2013

Abstract

In order to compute the log-likelihood for high dimensional spatial Gaussian models, it is necessary to compute the determinant of the large, sparse, symmetric positive definite precision matrix, \mathbf{Q} . Traditional methods for evaluating the log-likelihood for very large models may fail due to the massive memory requirements. We present a novel approach for evaluating such likelihoods when the matrix-vector product, $\mathbf{Q}\mathbf{v}$, is fast to compute. In this approach we utilise matrix functions, Krylov subspaces, and probing vectors to construct an iterative method for computing the log-likelihood.

1 Introduction

In computational and, in particular, spatial statistics, increasing possibilities for observing large amounts of data leaves the statistician in want of computational techniques capable of extracting useful information from such data. Large data sets arise in many applications, such as modelling seismic data acquisition (Buland and Omre (2003)); analysing satellite data for ozone intensity, temperature and cloud formations (McPeters et al. (1996)); or constructing global climate models (Lindgren et al. (2011)). Most models in spatial statistics are based around multivariate Gaussian distributions, which has probability density function

$$p(\mathbf{x}) = (2\pi)^{-k/2} \det(\mathbf{Q})^{1/2} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{Q}(\mathbf{x} - \boldsymbol{\mu})\right)$$

where the precision matrix \mathbf{Q} is the inverse of the covariance matrix. In this paper, we assume that the precision matrix is sparse, which essentially enforces a Markov property on the Gaussian random field. These models have better computational properties than those based on the covariance, and there are modelling reasons to prefer using \mathbf{Q} directly (Lindgren et al. (2011)). We note that Rue and Tjelmeland (2002) showed that it is possible to approximate general Gaussian random fields by Gaussian Markov random fields (GMRFs), that is Gaussian random vectors with sparse precision matrices.

Throughout this paper, we will consider the common Gauss-linear model, in which our data is a noisy observation of a linear transformation of a true random field, that is

$$\mathbf{y} = \mathbf{A}(\boldsymbol{\theta})\mathbf{x} + \boldsymbol{\epsilon}_1, \tag{1}$$

where the matrix $\mathbf{A}(\boldsymbol{\theta})$ models the observation of the true underlying field \mathbf{x} , known as the ‘forward model’, while $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_1^{-1})$ is the observation noise. In order to complete the model, we require a prior distribution on \mathbf{x} , which we take to be Gaussian with mean $\boldsymbol{\mu}$ and precision matrix $\mathbf{Q}_x(\boldsymbol{\eta})$,

and appropriate hyperpriors are given for the mean $\boldsymbol{\mu}$, the hyperparameters in the prior $\boldsymbol{\eta}$ and the hyperparameters in the forward model $\boldsymbol{\theta}$.

Given a set of data, we wish to infer $\boldsymbol{\eta}$, $\boldsymbol{\theta}$ and \mathbf{x} . The ways in which this is done can vary, but in the end, several determinant evaluations are needed. One way to do this is to alternately estimate \mathbf{x} and $\boldsymbol{\eta}$, $\boldsymbol{\theta}$ using the distributions $p(\mathbf{x}|\mathbf{y}, \boldsymbol{\eta}, \boldsymbol{\theta})$ and $p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$, updating each consecutively. That is

1. Find $\arg \max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}, \boldsymbol{\eta}, \boldsymbol{\theta})$
2. Find $\arg \max_{\boldsymbol{\eta}, \boldsymbol{\theta}} p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathbf{x}, \mathbf{y})$
3. Repeat until convergence.

In a Gauss-linear model, the first step involves a linear solve, while the second is an optimisation over the space in which $\boldsymbol{\eta}$, $\boldsymbol{\theta}$ reside. The distribution is

$$\begin{aligned} p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathbf{x}, \mathbf{y}) &\propto p(\mathbf{y}|\mathbf{x}, \boldsymbol{\eta}, \boldsymbol{\theta})p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathbf{x}) \\ &= p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\eta})p(\boldsymbol{\eta})p(\boldsymbol{\theta}) \end{aligned} \quad (2)$$

The log of the last line in (2) gives the objective function for the hyper-parameters in this case:

$$\Phi(\boldsymbol{\eta}, \boldsymbol{\theta}) = -\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\eta})p(\boldsymbol{\eta})p(\boldsymbol{\theta}) \quad (3)$$

We also have that $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\eta})$ as a function of \mathbf{x} is $\mathcal{N}(\boldsymbol{\mu}_p, \mathbf{Q}_p)$, where $\mathbf{Q}_p = \mathbf{Q}_x(\boldsymbol{\eta}) + \mathbf{A}(\boldsymbol{\theta})^T \mathbf{Q}_1 \mathbf{A}(\boldsymbol{\theta})$ and $\boldsymbol{\mu}_p = \mathbf{Q}_p^{-1}(\mathbf{Q}_x \boldsymbol{\mu} + \mathbf{A}^T \mathbf{Q}_1 \mathbf{y})$. The separated expression, $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\eta})$ in Φ , is usually preferable, but this is problem dependent.

Expanding (3), we get

$$\begin{aligned} \Phi(\boldsymbol{\eta}, \boldsymbol{\theta}) = & C - \log\left(\frac{1}{2} \det(\mathbf{Q}_1)\right) + \frac{1}{2}(\mathbf{y} - \mathbf{A}(\boldsymbol{\theta})\mathbf{x})^T \mathbf{Q}_1(\mathbf{y} - \mathbf{A}(\boldsymbol{\theta})\mathbf{x}) - \\ & \log\left(\frac{1}{2} \det(\mathbf{Q}_x(\boldsymbol{\eta}))\right) + (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{Q}_x(\mathbf{x} - \boldsymbol{\mu}) - \log p(\boldsymbol{\eta}) - \log p(\boldsymbol{\theta}). \end{aligned} \quad (4)$$

In this expression, the only term which is difficult to evaluate is $\log \det(\mathbf{Q}_x(\boldsymbol{\eta}))$, and it is needed for estimating the hyper-parameters, both for point estimates and for Gaussian- or Laplace-approximations of the hyper-parameters (see Carlin and Louis (2000) for details on such approximations). It is this evaluation and its use in optimisation we address in this article.

2 Determinant evaluations

The most common way to compute the log-determinant of a sparse precision or covariance matrix is to 1) reorder \mathbf{Q} to optimise for Cholesky factorisation, 2) perform a Cholesky factorisation of the reordered matrix $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$, 3) extract the diagonal entries of $\mathbf{l} = \text{diag}(\mathbf{L})$ and 4) set the log-determinant as $\log \det \mathbf{Q} = \sum_{j=1}^n 2 \log(l_j)$ (this comes from the identity $\det \mathbf{Q} = \det \mathbf{L} \det \mathbf{L}^T = (\det \mathbf{L})^2$). The algorithm takes very few lines to program, given a good sparse matrix sorting routine, such as METIS (Karypis and Kumar (1999)) and a fast sparse Cholesky factoring implementation, such as CHOLMOD (Davis and Hager (1999), Chen et al. (2008)). Problems occur, however, when there are massive amounts of fill-in in the Cholesky factorisation even after resorting the matrix in question. For a Gaussian Markov random field the dimensionality of the underlying parameter space affect the storage cost for computing the Cholesky factorisation. In \mathbb{R}^1 the cost is $\mathcal{O}(n)$, in \mathbb{R}^2 , $\mathcal{O}(n^{3/2})$ in \mathbb{R}^3 , $\mathcal{O}(n^2)$ (Rue and Held (2005)).

2.1 Alternative approximations

The starting point for an alternative, less memory intensive procedure for computing the log-determinant comes from the identity

$$\log \det \mathbf{Q} = \text{tr} \log \mathbf{Q} \quad (5)$$

In the symmetric positive definite case, this identity is proved noting that $\det \mathbf{Q} = \prod_{i=1}^n \lambda_i$ where $\{\lambda_i\}$ are the eigenvalues of \mathbf{Q} and that $\log \mathbf{Q} = \mathbf{V} \log(\mathbf{\Lambda}) \mathbf{V}^T$ with $\mathbf{\Lambda} = \text{diag}(\boldsymbol{\lambda})$ and \mathbf{V} contains the eigenvectors of \mathbf{Q} . Furthermore, $\text{tr}(\mathbf{V} \log(\mathbf{\Lambda}) \mathbf{V}^T) = \text{tr}(\mathbf{V} \mathbf{V}^T \log \mathbf{\Lambda}) = \text{tr} \log \mathbf{\Lambda}$, which gives the identity.

How can this be useful in computation, is the next question. A trivial observation shows that

$$\text{tr} \log \mathbf{Q} = \sum_{j=1}^n \mathbf{e}_j^T \log(\mathbf{Q}) \mathbf{e}_j$$

where $\mathbf{e}_j = (0, \dots, 1, \dots, 0)$ where the one is in position j . While it is cumbersome to carry out this computation, it is the basis for stochastic estimators of the log-determinant. Such estimators have been studied for the trace of a matrix, the trace of the inverse of a matrix and our case, the trace of the logarithm of a matrix. Details on this can be found in Hutchinson (1989) and Bai and Golub (1997). The Hutchinson stochastic estimator is described as follows: 1) Let \mathbf{v}_j , $j = 1, \dots, s$ be vectors with entries $P(v^k = 1) = 1/2$, $P(v^k = -1) = 1/2$ independently. 2) Let

$$\text{tr} \log \mathbf{Q} \approx \frac{1}{s} \sum_{j=1}^s \mathbf{v}_j^T \log(\mathbf{Q}) \mathbf{v}_j. \quad (6)$$

As this is a Monte Carlo method, it is possible to compute confidence regions for the estimators, using either Monte Carlo techniques or the Hoeffding inequality (Bai and Golub (1997)). The Hutchinson estimator was formulated for approximating $\text{tr} \mathbf{Q}^{-1}$ in which case, we replace the $\log \mathbf{Q}$ in (6) with \mathbf{Q}^{-1} .

The Hutchinson estimator requires a lot of random vectors \mathbf{v}_j to be sufficiently accurate for optimisation. The memory requirements are low, but we may have to wait an eternity for one determinant approximation. The question, then, can we choose the \mathbf{v}_j s in a clever way, so that we require far fewer vectors?

In recent publications, Bekas et al. (2007) and Tang and Saad (2010) explored the use of probing vectors for extracting the diagonal of a matrix or its inverse. In the first of these the diagonal of a sparse matrix is extracted, and it is relatively straightforward under mild assumptions to extract the diagonal. The second relies on approximate sparsity of the inverse, where the approximate sparsity pattern of \mathbf{Q}^{-1} is determined by a power of the original matrix, i.e. \mathbf{Q}^p . Assuming such a sparsity structure, it is possible to compute the probing vectors $\{\mathbf{v}_j\}_{j=1}^s$ by a neighbour colouring of the graph induced by \mathbf{Q}^p (see e.g. Culberson (1992) for a survey on the greedy graph colouring algorithms). A heuristic suggested in Tang and Saad (2010) to find the power, p in \mathbf{Q}^p is to solve $\mathbf{Q}\mathbf{x} = \mathbf{e}_j$ and find $p = \min\{d(l, j) \mid |x_l| < \epsilon\}$ where d gives the graph distance. In our case, we may compute $\log(\mathbf{Q})\mathbf{e}_j$ and use the same heuristic. A nice feature is that the probing vectors need not be stored, but may be computed cheaply on the fly. If we pre-compute them, they are sparse, and does not need much storage. Since what we need for each probing vector is $\mathbf{v}_j^T \log(\mathbf{Q}) \mathbf{v}_j$, we observe that the computation is highly parallel with low communication costs. Each node gets

one probing vector, and computes $\mathbf{v}_j^T \log(\mathbf{Q})\mathbf{v}_j$ and sends back the result. In essence, this leads to linear speedup with the amount of processors available with proportionality close to unity.

Next, we need to consider the evaluation of $\log(\mathbf{Q})\mathbf{v}_j$. The matrices we consider have real positive spectrum, and it is possible to evaluate $\log(\mathbf{Q})\mathbf{v}_j$ through Cauchy's integral formula,

$$\log(\mathbf{Q})\mathbf{v}_j = \oint_{\Gamma} \log(x)(z\mathbf{I} - \mathbf{Q})^{-1}\mathbf{v}_j dz,$$

where Γ is a suitable curve enclosing the spectrum of \mathbf{Q} and avoiding branch cuts of the logarithm. Direct quadrature over such curves can be terribly inefficient, but through clever conformal mappings, Hale et al. (2008) developed midpoint quadrature rules that converge rapidly for increasing number of quadrature points at the cost of needing estimates for the extremal eigenvalues of \mathbf{Q} . In fact, $\|\log \mathbf{Q} - f_N(\mathbf{Q})\| = \mathcal{O}(e^{-2\pi N/(\log(\lambda_{\max}/\lambda_{\min})+6)})$ with f_N as below. This essentially gives us the rational approximation

$$\log(\mathbf{Q})\mathbf{v}_j \approx f_N(\mathbf{Q})\mathbf{v}_j = \sum_{l=1}^N \alpha_l (\mathbf{Q} - \sigma_l \mathbf{I})^{-1} \mathbf{v}_j, \quad \alpha_l, \sigma_l \in \mathbb{C}. \quad (7)$$

In effect, we need to solve a family of shifted linear systems to approximate $\log(\mathbf{Q})\mathbf{v}_j$. How we compute $f_N(\mathbf{Q})\mathbf{v}_j$ is problem dependent, but in high dimensions, we usually have to rely on iterative methods, such as Krylov methods. A Krylov subspace, $\mathcal{K}_k(\mathbf{Q}, \mathbf{v})$ is defined by $\mathcal{K}_k(\mathbf{Q}, \mathbf{v}) = \text{span}\{\mathbf{v}, \mathbf{Q}\mathbf{v}, \mathbf{Q}^2\mathbf{v}, \dots, \mathbf{Q}^k\mathbf{v}\}$ and a thorough introduction to the use and theory of Krylov methods can be found in Saad (2003). Which Krylov method we use is highly dependent on the quality and performance potential preconditioners for the matrix \mathbf{Q} .

While the Krylov method of choice is problem dependent, there are ones that are particularly well suited to compute the approximation in (7). These methods are based on the fact that $\mathcal{K}_k(\mathbf{Q}, \mathbf{v}) = \mathcal{K}_k(\mathbf{Q} - \sigma_l \mathbf{I}, \mathbf{v})$ and after some simple algebra, we obtain the coefficients for the shifted systems without computing new matrix-vector products, see Jegerlehner (1996) and Frommer (2003) for details. We have employed the method CG-M in Jegerlehner (1996) our computations. One possible difficulty in employing the method is that we have complex shifts - this is remedied by using a variant, Conjugate Orthogonal CG-M (COCG-M), which entails using the conjugate symmetric form $(\bar{\mathbf{x}}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$ instead of the usual inner product $(\mathbf{x}, \mathbf{y}) = \bar{\mathbf{x}}^T \mathbf{y}$ in the Krylov iterations. See van der Vorst and Melissen (1990) for a description of the COCG method. In practice, little complex arithmetic is needed, since the complex, shifted coefficients are computed from the real ones obtained by the CG method used to solve $\mathbf{Q}\mathbf{x} = \mathbf{y}$.

3 Examples

In this section we explore how optimisation fares under different conditions. For doing this, we assume essentially the simplest possible model, but we plan use the outlined approach on a seismic case in the future. The model we assume is the SPDE $\tau(\kappa - \Delta)u = \mathcal{W}$, in 2D, which we observe directly. We will explore how optimisation works (on τ, κ) for different κ s and different distance colouring of the graph. Note that the number of colours needed is essentially independent of the granularity of the discretisation: a fine grid yields approximately the same number of colours as a coarse grid. The initial suspicion is that for small κ s, corresponding to long range will be harder to optimise in the following sense: we need more Krylov iterations for the COCG-M routine

to converge and we need a larger distance colouring to cover the increasing range, resulting in more probing vectors. We use a modified Newton method for optimisation and compare using the exact determinant to using the approximation outlined above. For the COCG-M method, we use a relative tolerance of 10^{-3} for computing $\log(\mathbf{Q})\mathbf{v}_j$. Note that a prior on the parameters will stabilise the results as usual. The results are given in Table 1.

Table 1: Optimisation of (κ, τ) for different distance colourings

	Exact	2-dist	4-dist	6-dist	8-dist	10-dist
$\kappa = 1$	(0.927, 1.015)	(1.06, 0.98)	(0.933, 1.013)	(0.927, 1.015)
$\kappa = 0.5$	(0.455, 1.010)	(0.605, 0.961)	(0.471, 1.005)	(0.457, 1.009)	(0.455, 1.010)	...
$\kappa = 0.1$	(0.0842, 1.003)	(0.208, 0.940)	(0.122, 0.984)	(0.0983, 0.996)	(0.0891, 1.000)	(0.0859, 1.002)
$\kappa = 0.05$	(0.0398, 1.001)	(0.138, 0.941)	(0.0762, 0.980)	(0.0567, 0.992)	(0.0475, 0.997)	(0.0434, 0.999)
$\kappa = 0.01$	(0.00644, 0.998)	(0.0565, 0.947)	(0.0292, 0.978)	(0.197, 0.987)	(0.0143, 0.992)	(0.0117, 0.994)

In Table 1, ... indicates that the optimisation yields the same as the previous entry. We see that as we increase the graph neighbourhood in our colourings, we get results closer and closer to that of using the exact determinant. We also see that the estimates are monotone: κ decreases with larger distance colourings and τ increases. Lastly, we see that some of the estimates are better when using the approximation. This should not necessarily be taken as a good sign, as it may only be a result of approximation errors.

As increasing k in k -distance colourings yields better and better approximations, one could be lead to using a "large" k in the entire optimisation, whatever method one uses to determine k . Our results indicate, however, that we only need very good approximations in the last iterations of the optimisation procedure. In effect, we may use 2-distance colourings in the beginning, and go to 5- or larger -distance colourings in the last steps. Computing the colourings is cheap and one colouring only requires the storage of one vector, so we may store a couple of different colourings and use them as required in the optimisation procedure.

Lastly, we present an example that cannot be done using black-box Cholesky factorisations. Namely, a 3-D version of the model above with $\kappa = 0.5$ and 2 million discretisation points. We use a 1-distance colouring for the first iterations and increase to 2- and 6-distance colouring in the last iterations. This will give us temporary k -distance estimates which we also give. The estimates, progressively from 1-, 2 and 6-distance colouring were (7.627, 0.382), (1.401, 0.801), and (0.561, 0.988). It took 24 hours to complete the optimisation. From this we conclude - the method is slow, but it can be used for parameter estimation in high-dimensional problems where other alternatives are impossible due to memory limitations. We must, however, be careful so that we have enough colours to capture the essentials of the determinant. The estimated memory use for using Cholesky factorisation in the determinant evaluations is 155 Gb with METIS sorting of the graph and much higher without. Few computing servers have this amount of memory on a node. The memory consumption for the approximation was 3 Gb at maximum, and even this may be lowered quite a lot with some clever memory management. In a computing cluster, the time for computing this optimisation would have been much lower due to linear speedup vs. number of nodes.

We have tried the approximation for different types of matrices, and what we found is that the examples above are among the toughest to do determinant approximations on. Since we can do reasonable approximations for this class, we expect that these likelihood evaluations will work for

a large class of precision matrices in use in statistics.

4 Discussion

We have showed that the determinant approximations discussed shows promise for likelihood evaluations in models where we cannot perform Cholesky factorisations or Kronecker decompositions of the precision matrices. This may prove useful in high dimensional models where approximate likelihoods are not sufficient for accurate inference. It remains to utilise the approximations on a real world dataset.

References

- Bai, Z. and Golub, G. (1997). Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices. *Annals of Numerical Mathematics*, 4:29–38.
- Bekas, C., Kokiopoulou, E., and Saad, Y. (2007). An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11-12):1214–1229.
- Buland, A. and Omre, H. (2003). Bayesian linearized avo inversion. *Geophysics*, 68:185–198.
- Carlin, B. and Louis, T. (2000). *Bayes and Empirical Bayes methods for data analysis*. CRC Press.
- Chen, Y., Davis, T., Hager, W., and Rajamanickam, S. (2008). Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22.
- Culberson, J. (1992). Iterated greedy graph coloring and the difficulty landscape. Technical report, University of Alberta.
- Davis, T. and Hager, W. (1999). Modifying a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 20(3):606–627.
- Frommer, A. (2003). BiCGStab (l) for families of shifted linear systems. *Computing*, 70(2):87–109.
- Hale, N., Higham, N. J., and Trefethen, L. N. (2008). Computing A^α , $\log(A)$ and related matrix functions by contour integrals. *SIAM Journal of Numerical Analysis*, 46:2505–2523.
- Higham, N. J. (2008). *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
- Hutchinson, M. (1989). A stochastic estimator of the trace of the influence matrix for $\{L\}$ aplacian smoothing splines. *Commun. Stat. Simula.*, 18:1059–1076.
- Jegerlehner, B. (1996). Krylov space solvers for shifted linear systems. *arXiv.org, arXiv:hep-lat/9612014v1*, NA:NA.
- Karypis, G. and Kumar, V. (1999). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359.

- Lindgren, F., Lindstrøm, J., and Rue, H. (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: The SPDE approach. *Journal of the Royal Statistical Society, Series B*, 5, to appear.
- McPeters, R., Aeronautics, U. S. N., Scientific, S. A., and Branch, T. I. (1996). *Nimbus-7 Total Ozone Mapping Spectrometer (TOMS) data products user's guide*. NASA, Scientific and Technical Information Branch.
- Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields*. Chapman & Hall.
- Rue, H. and Tjelmeland, H. (2002). Fitting gaussian markov random fields to gaussian fields. *Scandinavian Journal of Statistics*, 29:31–49.
- Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems, 2nd Ed.* SIAM.
- Tang, J. and Saad, Y. (2010). A probing method for computing the diagonal of the matrix inverse. Technical report, Minnesota Supercomputing Institute for Advanced Computational Research.
- van der Vorst, H. and Melissen, J. (1990). A Petrov-Galerkin type method for solving $Ax=b$, where A is symmetric complex. *Magnetics, IEEE Transactions on*, 26(2):706–708.